Avoiding Communication in Matrix Multiplication

SURF Math Team
Anthony Chen, Mason Haberle, Jon Hillery, Rahul Jain

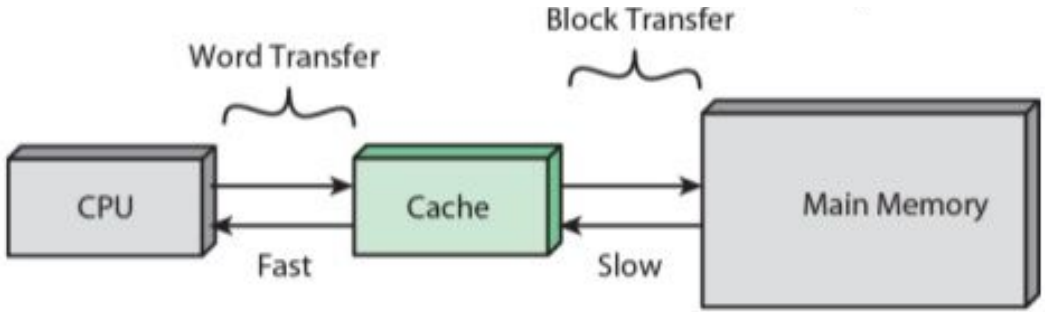
Communication Costs

We want to make programs take less time and energy.

Past Goal: Write algorithms to minimize calculations and arithmetic.

Now: Arithmetic is fast! Moving around information is still slow.

Current Goal: Write algorithms to minimize movement of information.

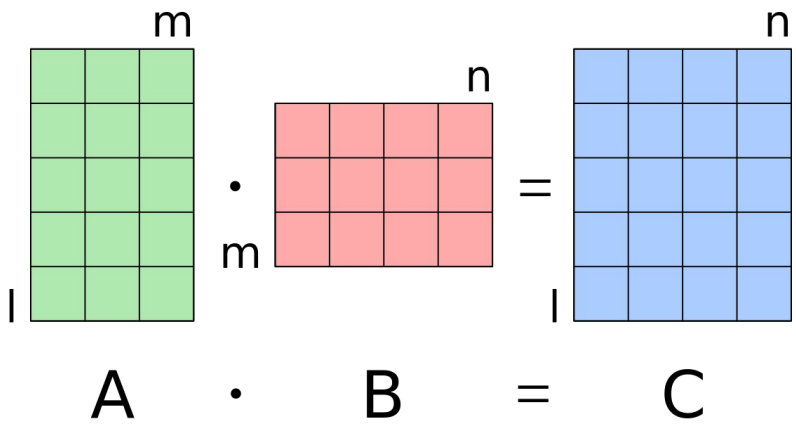


Numerical Linear Algebra

Important for just about any computing application!

Machine learning, graphics, database manipulation, scientific computing, search

Basic operations such as matrix multiplication and matrix decompositions



Multiplying Matrices - Calculation

We want a program to multiply two N x N matrices.

If we want to find C = A x B, the following code does the calculation:

```
for i = 1 to n
  for j = 1 to n
    for k = 1 to n
      C[i,j] = C[i,j] + A[i,k] * B[k,j]
```

In this code, we do about $2N^3$ arithmetic floating-point operations total. We do:

- N^3 multiplications of numbers
- N^3 additions of numbers

We say the program does $O(N^3)$ arithmetic operations.

Multiplying Matrices - Communication

With this algorithm, how much information do we need to communicate?

If there are M spots in fast memory, then to multiply N x N matrices

we need to do at least $\frac{N^3}{8\sqrt{M}} - M$ communications.

We can write that the number of communications is $\Omega\left(N^3/\sqrt{M}\right)$.

More memory means less communication.

This same result has been proven in two very different ways.

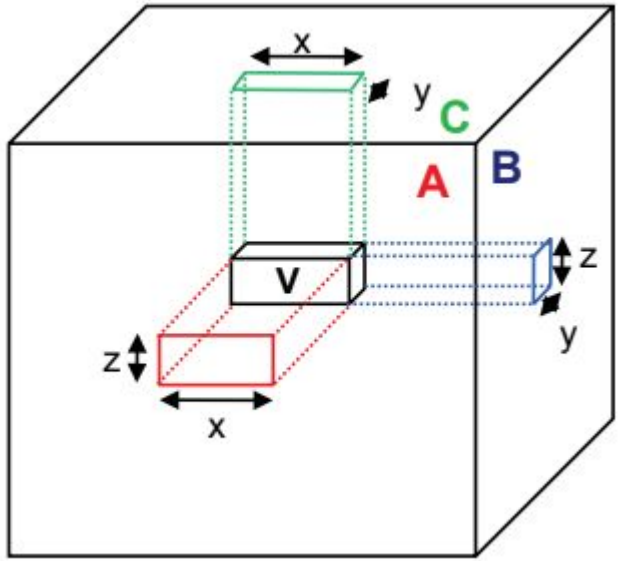
Proof Tool: The Loomis-Whitney Inequality

Say V is a finite set of points in $\mathbb{Z}^3$
i.e. a set of integer triples (i, j, k).

Then if X = {(j, k) : (i, j, k) is in V} is the set with the x-coordinate removed,
Y the set with the y-coordinate removed, and
Z the set with the z-coordinate removed,

and if |V|, |X|, |Y|, and |Z| denote the number of points in each set,

We have $|V| \leq \sqrt{|X| \cdot |Y| \cdot |Z|}$.



X is the blue rectangle,
Y is the red rectangle,
Z is the green rectangle.

Proof Tool: The Red-Blue Pebble Game

We can model a computation using a directed acyclic graph (DAG).

Each node is a calculation.
Each arrow is a dependency.

We use pebbles to say whether a piece of data is in fast or slow memory.

Blue pebbles represent slow memory.
There are arbitrarily many.

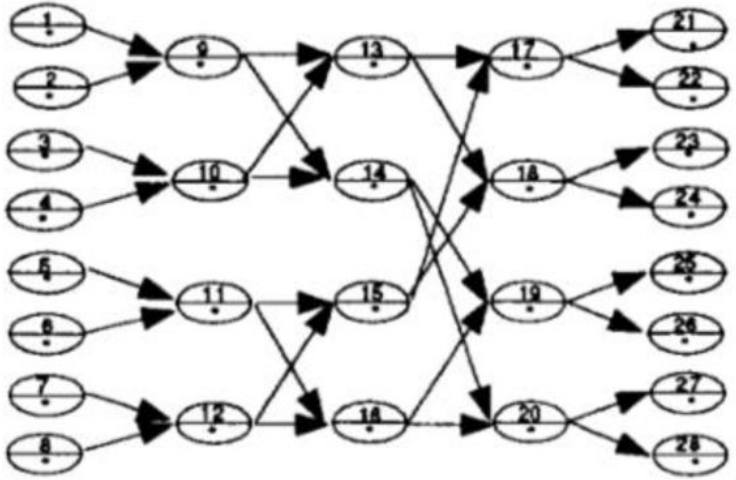


Figure 1. DAG for Fast Fourier Transform (FFT) [7]

Red pebbles represent fast memory. There are only M.

Red-Blue Pebble Game Rules

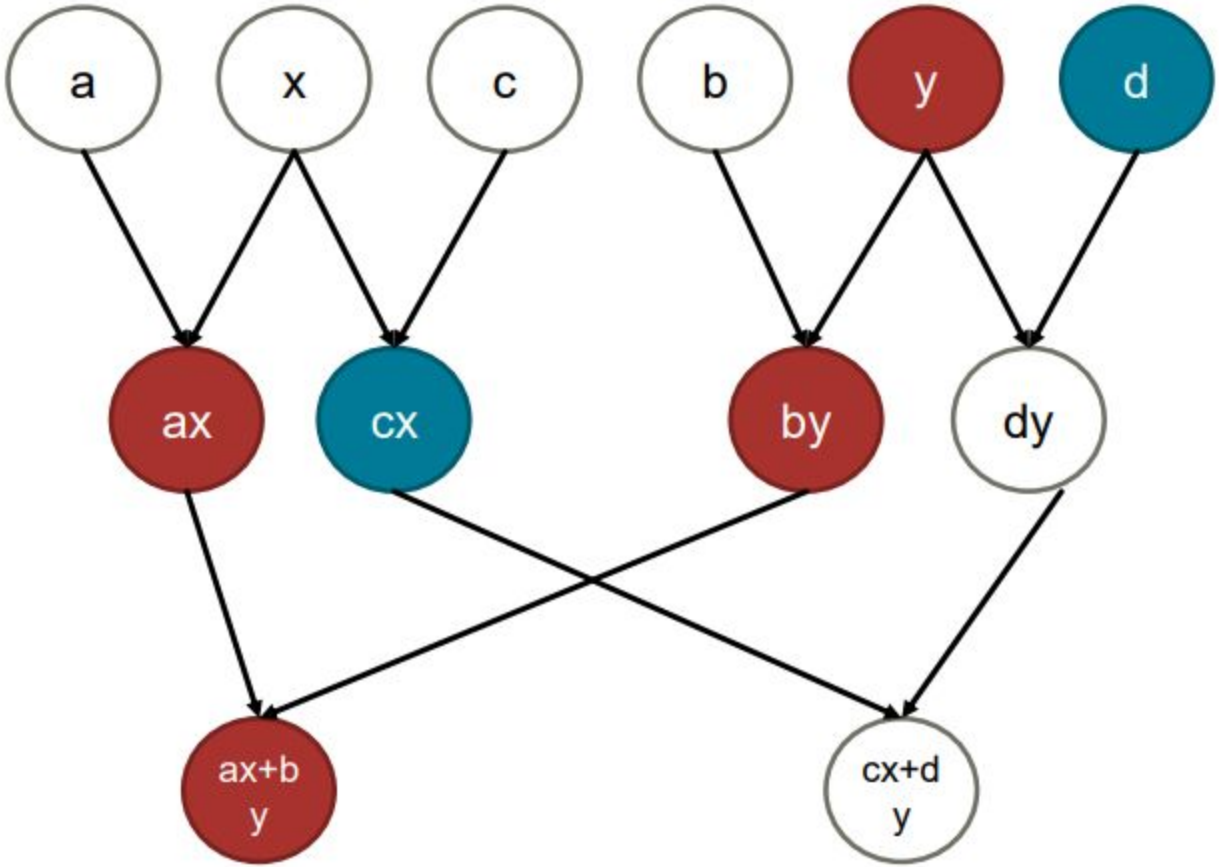
Start: Blue pebbles on all inputs. Goal: blue pebbles on all outputs.

The moves you can make are:

1. (Load) Put a red pebble on a node which has a blue pebble.
2. (Store) Put a blue pebble on a node which has a red pebble.
3. (Compute) Put a red pebble on a node whose parents all have red pebbles.
4. (Delete) Remove any pebble.

How many loads/stores do you have to do, if there are only M red pebbles?

This is the communication cost.


a
x
c
b
y
d
ax
cx
by
dy
ax+b
y
cx+d
y

Steps of the Proof

1. Split the computation process into segments, each with M communications.
2. Find how much computing you can do in a segment.
3. Find how many segments you need.

Step 1: Split into Segments

Split up the computation process into S segments.

In a segment, we allow exactly M communications between fast and slow memory.

So that's M loads from memory (e.g. finding entries of the matrices A or B)

or M stores to memory (e.g. filling in the entries of the matrix C = A x B).

Step 2: Calculations Per Segment

In the current segment, let V be the set of triples (i, j, k) such that we compute the following line of code during the segment:

```
C[i,j] = C[i,j] + A[i,k] * B[k,j]
```